Cours de Compilation

Prof. M.D. RAHMANI

Plan du cours

I. Généralités sur les compilateurs.
II. Notion de la Grammaire
III. Analyse lexicale
IV. Le langage FLEX
V. Analyse syntaxique
VI. Le langage BISON.

Chapitre I:
Généralités sur les compilateurs

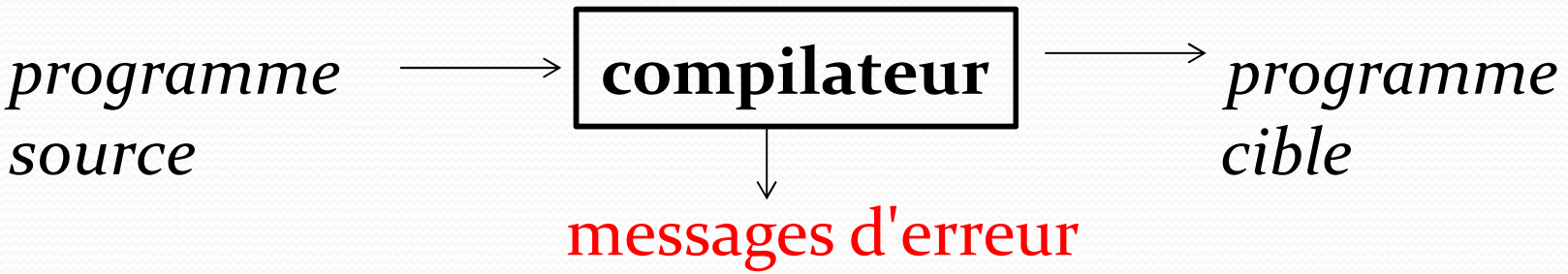
1. Les compilateurs,
2. Les phases d'analyse du programme source,
3. Les phases de synthèse,
4. Regroupement des phases
5. Chargeurs et relieurs,
6. Interprètes

1- Les compilateurs

1.1- **Définition**:

*Un compilateur est un programme qui **lit** un programme écrit dans un premier langage (le langage source) et le **traduit** en un programme équivalent dans un autre langage (le langage cible).*

Le compilateur doit aussi vérifier que le programme a un certain sens et signaler les erreurs qu'il détecte.



1- Les compilateurs

Il y'a deux parties dans la compilation: l'***analyse*** et la ***synthèse***.

- ***La partie analyse*** partitionne le programme source en ses constituants et en crée une représentation intermédiaire.

- ***La partie synthèse*** construit le programme cible à partir de cette représentation intermédiaire.

1- Les compilateurs

1.2- **Arbre abstrait**:

Pendant l'analyse, les opérations spécifiées par le programme source sont déterminées et conservées dans une structure hiérarchique appelée **arbre abstrait**.

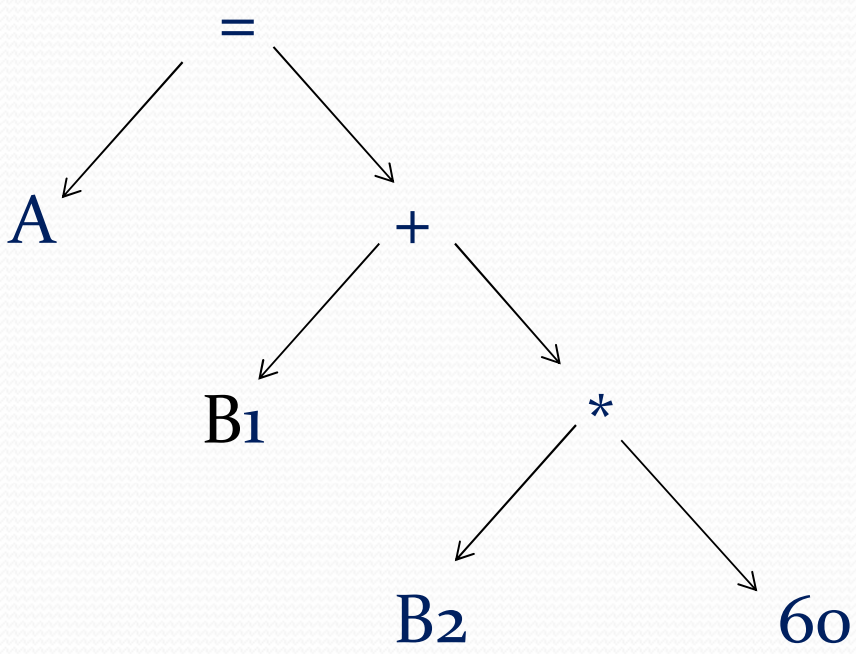
*Un arbre abstrait est constitué de **nœuds** qui représentent les **opérations** et les **fils des nœuds** qui représentent les **arguments** des opérations*

1- Les compilateurs

Exemple

Soit l'instruction: A = B1 + B2 * 60

Son **arbre abstrait** est:



2- Les phases d'analyse

L'analyse est constituée de 3 parties:

1. **L'analyse lexicale** (linéaire): le flot de caractères formant le programme source est **lu** de gauche à droite et **groupé** en *lexèmes* (mots), qui sont des suites de caractères ayant une signification collective.

2. **L'analyse syntaxique** (grammaticale): les unités lexicales sont regroupés hiérarchiquement dans des collections imbriquées (phrases) ayant une signification collective.

3. **L'analyse sémantique**: contrôle pour s'assurer que l'assemblage des constituants du programme a un sens.

Exemple d'analyse

Soit l'instruction: A = B1 + B2 * 60

2.1- **L'analyse lexicale**:

1. L'identificateur "A"
2. Le symbole d'affectation "="
3. L'identificateur "B1"
4. Le signe "+"
5. L'identificateur "B2"
6. Le signe de multiplication "*"
7. Le nombre "60"

Remarque: les blancs qui séparent les mots sont éliminés.

Exemple d'analyse (suite)

2.2- **L'analyse syntaxique**:

Les structures grammaticales sont représentées par un arbre syntaxique et généralement exprimées par des règles récursives.

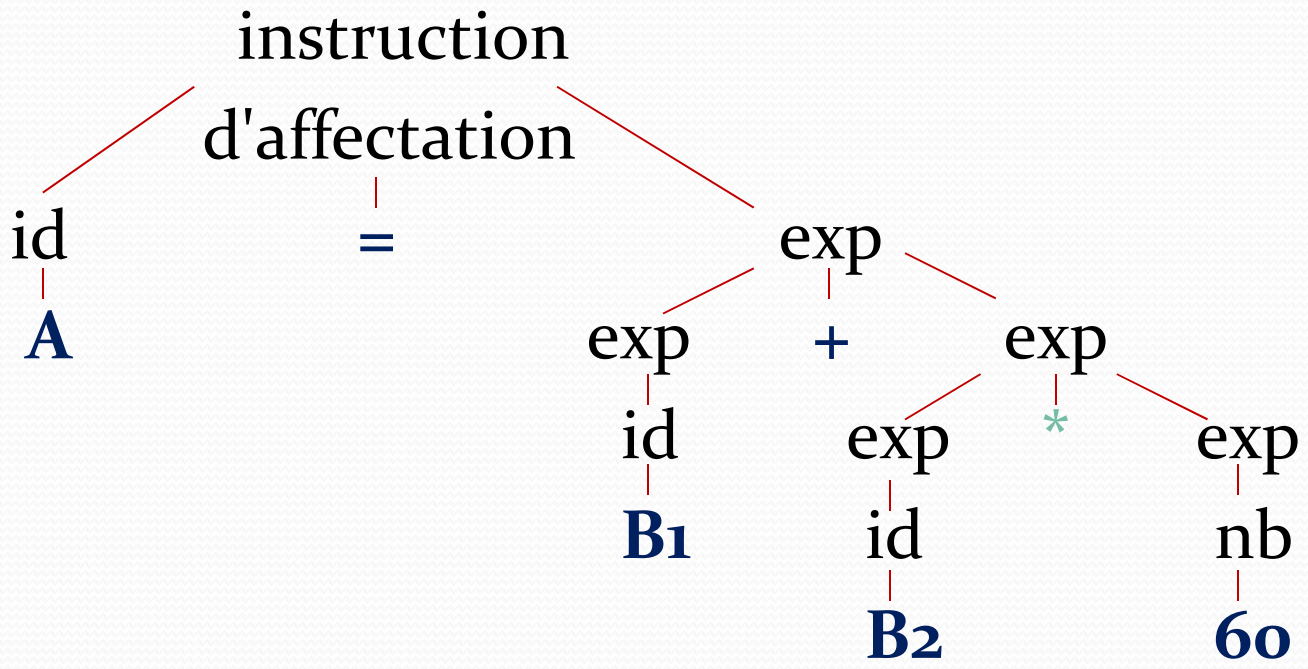
Les règles de grammaires:

- règles de base non récursives:
  1. *Tout identificateur est une expression.*
  2. *Tout nombre est une expression.*
- règle récursive:
  3. *Si **e1** et **e2** sont des expressions, **e1**+**e2**, **e1*****e2** et **(e1)** sont aussi des expressions.*

Exemple d'analyse (suite)

4. *Une instruction peut être de la forme: id = exp*

L'arbre syntaxique correspondant:



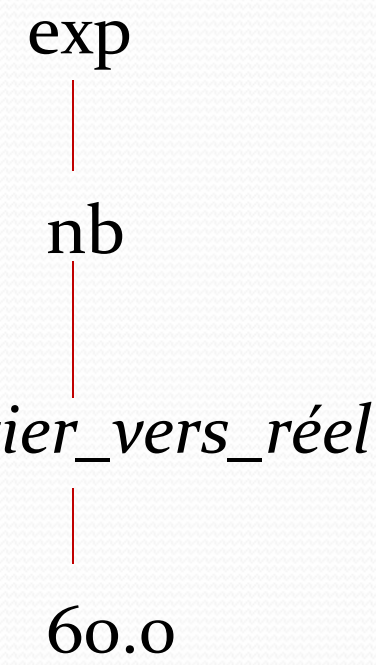
Remarque: L'arbre abstrait est une forme compacte de l'arbre syntaxique

Exemple d'analyse (suite)

2.3. **L'analyse sémantique**:

Un constituant important est le contrôle su type.

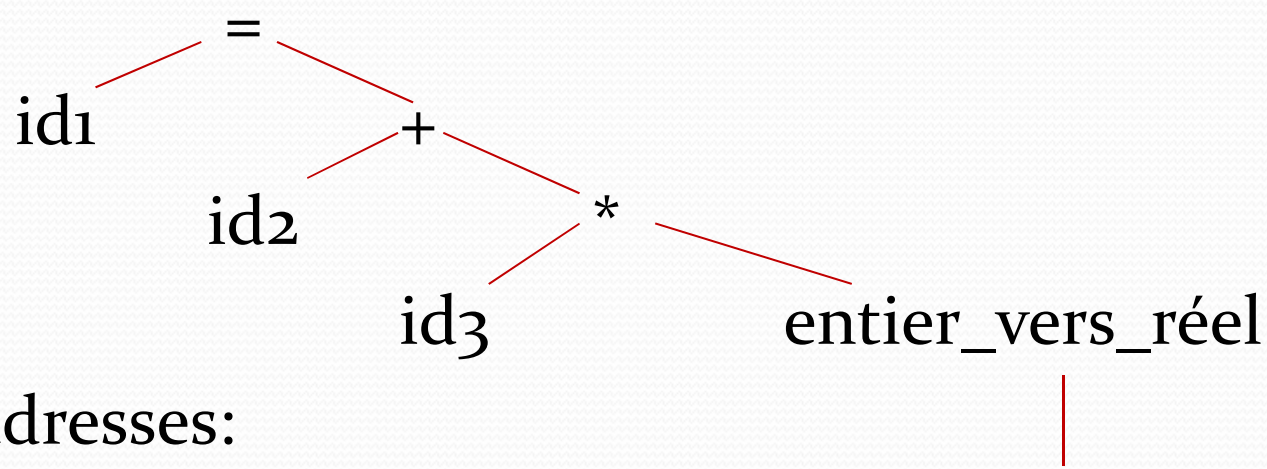
Si nous déclarons A, B1 et B2 des réels, alors nous devons convertir l'entier 60 en réel 60.0



Exemple d'analyse (suite)

2.4. **Le code intermédiaire:**

L'arbre abstrait:



Le code à 3 adresses:

```
temp1 := entier_vers_réel(60);
temp2 := id3 * temp1;
temp3 := id2 + temp2;
id1   := temp3;
```

3- Les phases de synthèse

3.1- **Optimisation du code**:

Amélioration du code intermédiaire pour que le code final s'exécute plus rapidement et utilise le minimum de mémoire.

```
temp1 := id3 * 60.0;
id1 := id2 + temp1;
```

Exemple de synthèse

3.2- **Génération du code**:

Production du code cible en langage d'assemblage.

L'aspect crucial est l'assignation des variables aux registres du processeur.

Si nous utilisons 2 registres *R1* et *R2*, la traduction pourrait être :

```
MOVF  R2,id3
MULF  R2,#60.0
MOVF  R1,id2
ADDF  R1,R2
MOVF  id1,R1
```

F: nombre en virgule flottante

#: 60.0 doit être traité comme une constante

4- Regroupement des phases

L'étude précédente traite de l'organisation logique d'un compilateur. Souvent on regroupe les phases en 2 parties:

4.1- **La partie frontale**: constituée des phases qui dépendent principalement du langage source:

- l'analyse lexicale,
- l'analyse syntaxique,
- l'analyse sémantique,
- la création de la table des symboles,
- la production du code intermédiaire
- elle inclut le traitement des erreurs associées à chacune de ces phases.

4- Regroupement des phases

4.2- La partie finale:

Elle est constituée des phases qui dépendent de la machine cible.

- Optimisation du code intermédiaire,
- production du code final,
- gestion de la table des symboles,
- traitement des erreurs.

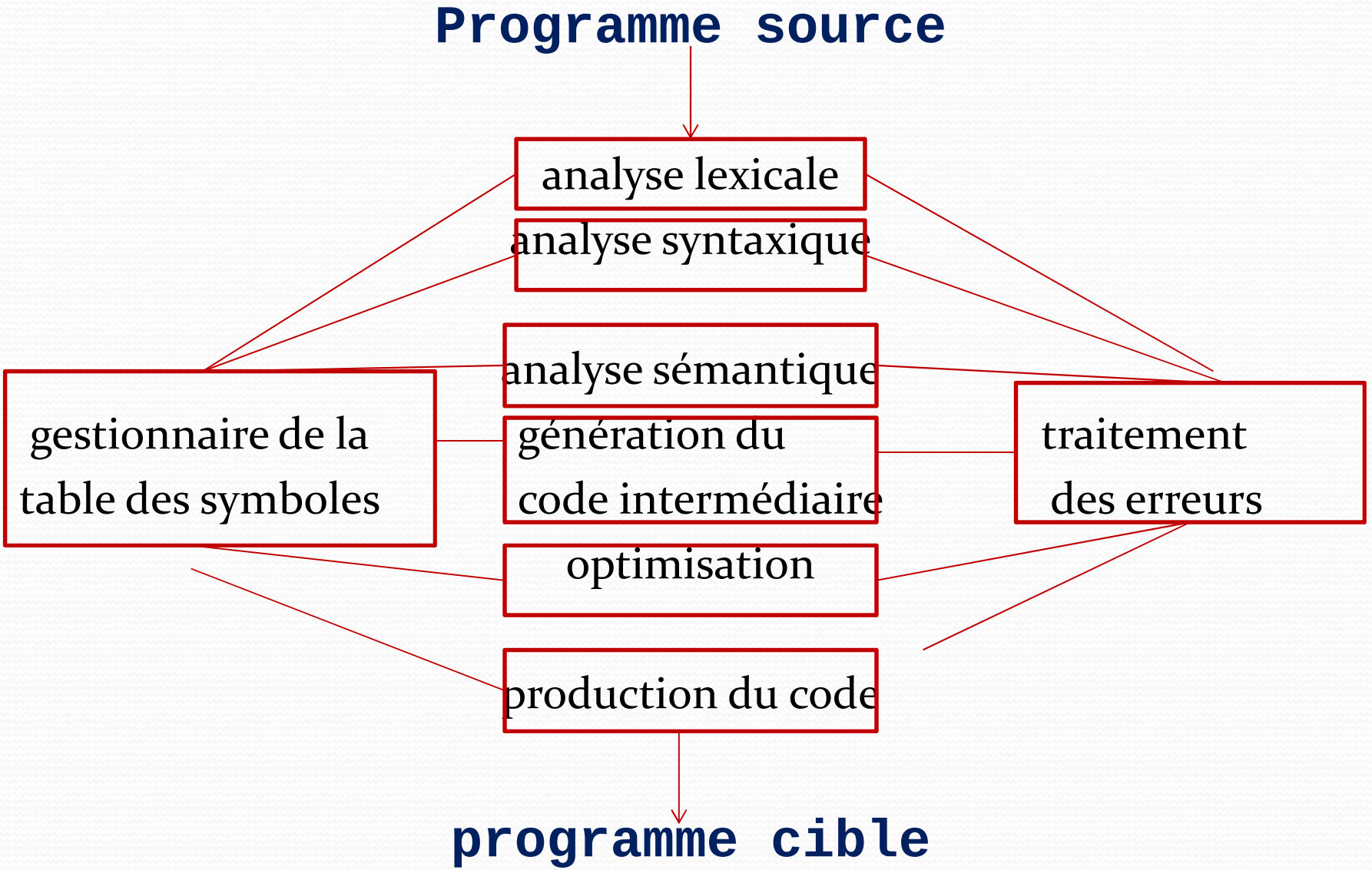
Les compilateurs

Conceptuellement un compilateur opère en 6 phases,

Chacune transforme le programme source d'une représentation en une autre.

En outre 2 phases interagissent avec les 6 phases

Les compilateurs



Les compilateurs

Les différentes phases		Outils utilisés
Phases d'analyse	analyse lexicale	- expressions régulières - automates à états finis
	analyse syntaxique	- grammaires - automates à piles
	analyse sémantique	- traduction dirigée par la syntaxe
Phases de synthèse	code intermédiaire	- traduction dirigée par la syntaxe
	optimisation	
	production du code	
Traitements parallèles	table des symboles	
	traitement des erreurs	

Les compilateurs

Gestion de la table des symboles:

Une table des symboles est une structure de données contenant un enregistrement pour chaque identificateur, muni de champs pour ses attributs (*emplacement mémoire, son type, sa protée..*).

Pour les fonctions, la table des symboles garde le *nombre* et *les types de ses arguments*, *le mode de passage de chacun d'eux et la valeur de retour*.

Les compilateurs

Gestion de la table des symboles:

indice	nom	adresse	…	
1	A			
2	B1			
3	B2			
4	…			

Les compilateurs

Traitement des erreurs:

Chaque phase peut rencontrer des erreurs.

Après avoir détecté une erreur, une phase doit la traiter de telle façon que la compilation puisse continuer et que d'autres erreurs dans le programme puissent être détectées.

5- **Chargeurs et relieurs** (édition de liens)

Charger les fichiers et les fonctions des bibliothèques et les relier pour la production du code exécutable.

6- Les interprètes

Au lieu de produire un programme cible, un interprète effectue lui même les opérations spécifiées par le programme source.

Un **interprète** est un outil ayant pour tâche d'analyser, de traduire et d'exécuter les programmes.

On utilise souvent des interprètes pour exécuter les langages de commande.

exemples:

- Le *shell* d'Unix
- Les langages de script: PHP, Perl

Certains langages ont deux versions, une compilée et une autre interprétée.

exemples: caml

Bibliographie:

Livres de référence:

1- A. Aho, M. Lam, R. Sethi et J. Ullman,

"*Compilateurs: principe, techniques et outils*", Pearson Education.

2- J. Menu, *" Compilateurs avec C++"*, Addison - Wesley.

3- R. Wilhelm et D. Maurer, *"Les compilateurs: théorie, construction, génération"*, Masson

4- J.E.F. Friedel, "*Maîtrise des expressions régulières*", O'Reilly

Webographie:

1- Compilateurs avec C++, Jacques Menu

http://cui.unige.ch/~menu/CompilateursAvecC++.pdf

2- Cours de techniques de compilation, J.Bonneville Documentation Lex-Flex

http://users.polytech.unice.fr/~dedale/cours/compilation/Lex-HowTo/

3- Cours de compilation, Luc Maranget

http://www.enseignement.polytechnique.fr/profs/informatique/Luc.Maranget/compil/poly/index.html

4- Cours de compilation, ENS de Lyon, Christophe Alias, cours, TD, TP, examens et partiels

http://perso.ens-lyon.fr/christophe.alias/compilation_ens.html

Webographie:

5- Techniques et outils pour la compilation, H. Garreta

Faculté des Sciences de Luminy - Université de la Méditerranée

http://henri.garreta.perso.luminy.univmed.fr/Polys/PolyCompil.pdf

6- Introduction à la compilation, Yann Régis-Gianas

Université Denis Diderot – Paris 7

http://www.pps.univ-paris-diderot.fr/~yrg/compil/compilation-slides-cours-1.pdf

7- Compilation, théorie des langages, université de Bretagne occidentale

http://www.lisyc.univ-brest.fr/pages_perso/leparc/Etud/Master/Compil/Doc/CoursCompilation.pdf

8- Cours de compilation, J. Ferber

http://www.lirmm.fr/~ferber/Compilation/compil1.htm

Outils

Flex et *bison*: http://sourceforge.net/projects/winflexbison/

Nous aurons un fichier compressé: **`win_flex_bison-latest`** à décompresser.

ou http://sourceforge.net/projects/gnuwin32/files/bison/2.4.1/bison-2.4.1-setup.exe/download

JFlex: logiciel http://jflex.de/ , manuel: http://jflex.de/manual.html

et *Cup*: http://www2.cs.tum.edu/projects/cup/

Dev-C++: http://www.commentcamarche.net/download/telecharger-59-dev-c

Nous aurons un fichier exécutable: **`Dev-Cpp_5.9.2_TDM-GCC_4.8.1_Setup`**

Geany: http://www.geany.org/ ,

manuel: http://www.geany.org/Documentation/Manual